Express Mail No. EL821157938US
Docket No. STL920000098US1
Firm No. 0055.0043

METHOD, SYSTEM, AND PROGRAM FOR PERFORMING WORKFLOW RELATED OPERATIONS

RELATED APPLICATIONS

This application is related to the following copending and commonly assigned patent applications, which are incorporated herein by reference in their entirety:

"Method, System, and Program for Generating a Workflow", having attorney docket no. STL920000094US1 and filed on June 28, 2001;

"Method, System, and Program for Using Objects In Data Stores During Execution of a Workflow", having attorney docket no. STL920000095US1 and filed on June 28, 2001;

"Method, System, and Program for Executing a Workflow", having attorney docket no. STL920000099US1 and filed on June 28, 2001;

"Method, System, and Program for Maintaining Information in Database Tables and Performing Operations on Data in the Database Tables", having attorney docket no. STL920000097US1 and filed on the same date hereof.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present invention relates to a method, system, and program for performing workflow related operations.

2. Description of the Related Art

[0002] A workflow program allows businesses and other organizations to define their business operations as a computer model known as a workflow. A workflow defines a series of processes to be performed by users at a client computer. The user activities at the client computers may involve updating an electronic form, reviewing information, etc. After one user in the workflow performs a specified action, the work item or other

information is then routed to one or more further nodes where further action may be taken. For instance, an on-line purchase of a product may involve numerous steps, such as receiving the customer order, routing the customer order to the credit department to process the bill and then routing the order to the shipment department to prepare the shipment. Once the shipment is prepared, the product may be shipped and information on the purchase is then transferred to the customer service department to take any further action. Each of these processes may be defined as nodes in a workflow. A workflow program would then route the customer order to the business agents designated to handle the job. For instance, the initial order would be received by the order department and then routed to a person in shipping and billing. Once the bill and package are prepared, a further invoice may be forwarded to shipping. After shipping sends the package, the shipping agent may then enter information into the invoice and forward the electronic invoice to customer service for any follow up action.

[0003] A workflow is designed using workflow software, such as the International Business Machines Corporation (IBM) MQSeries** Workflow software product. A process modeler is a person that analyzes the business operations, determines how the information related to the operations is routed electronically to client computers, and then defines a workflow model of the operations. The workflow model may be coded in the FlowMark Definition Language (FDL). The workflow model is then imported into a Runtime program that verifies and translates the workflow model into a process template. An instance of the process template can then be invoked to automates the sequence of events defined by the model.

[0004] Current workflow programs provide an application program interface (API) for client applications to use to interact with the workflow engine and perform workflow related functions and operations. In prior art implementations where the API is implemented in Java**, a Java Native Interface (JNI) wrapper is provided to translate the Java APIs to the native code of the workflow engine. The workflow engine then executes the native code to perform the function specified by the API. In addition, a

database is usually provided to store workflow related information and metadata. Java APIs are also provided to access workflow metadata in the workflow database. Thus, the client program would issue Java APIs to interact with the workflow engine and separately call other APIs to interact with the workflow database to access and manipulate workflow metadata.

[0005] The programming of the JNI interface are very difficult and cumbersome to code and debug. Further, the JNI interface developed for one vendor workflow engine is not reusable because it is closely coupled with the specific workflow engine implementation and cannot be reused with other vendor workflow engine implementations. In fact, the software developer would have to recompile every JNI interface based on the compiler and client/server architecture used for different vendor workflow engines.

[0006] For these reasons, there is a need in the art to provide improved designs of workflow programs.

SUMMARY OF THE PREFERRED EMBODIMENTS

[0007] Provided is a method, system, and program for performing workflow related operations. An application programming interface call (API) is received to perform a workflow related operation. A determination is made of at least one stored procedure call associated with the received API. The determined at least one stored procedure is then called to cause the execution of one determined stored procedure on a database server to perform the workflow related operation of the API.

[0008] In further implementations, at least one stored procedure call invokes one stored procedure in the database server that includes native workflow server code to communicate to a workflow server to execute to perform the workflow related operation of the API and at least one stored procedure call invokes one stored procedure in the database server that includes database statements executed by a database program to access workflow related metadata for the API.

[0009] Further provided is a method, system, and program for performing workflow related operations at a database server in communication with a client. At least one call to one stored procedure is received at the database server associated with an application programming interface call (API) invoked from the client to perform a workflow related operation. The database server executes the at least one called stored procedure to perform the workflow related operation of the API.

[0010] In still further implementations, at least one call to one stored procedure comprises a first and second stored procedure calls to first and second stored procedures, respectively, on the database server to implement the workflow related operation of the called API.

[0011] The described implementations provide a workflow design in a client/server environment that utilizes stored procedures on a database server to perform the workflow related operations associated with a workflow related API. The stored procedures may include native workflow server code to send to the workflow server to perform the workflow related operations specified by the API and/or access workflow metadata in a workflow database.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates a workflow computing environment in which aspects of the invention are implemented;

FIG. 2 illustrates logic performed by a workflow server to execute a workflow in accordance with implementations of the invention;

FIG. 3 illustrates an architecture of object oriented classes for implementing a workflow in accordance with implementations of the invention;

FIGs. 4 and 5 illustrate logic to utilize the methods and objects from the object oriented class architecture of FIG. 3 to execute a workflow in accordance with implementations of the invention;

FIG. 6 illustrates an architecture of a workflow computing environment in accordance with implementations of the invention; and

FIGs. 7 and 8 illustrate logic implemented in the architecture of FIG. 6 to perform workflow related operations in accordance with implementations of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0013] In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several embodiments of the present invention. It is understood that other embodiments may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

[0014] FIG. 1 illustrates a workflow environment implementation in which the invention is realized. A workflow engine 2 includes a runtime database 4 and a workflow server 6, such as the IBM MQSeries Workflow server. The workflow server 6 is capable of transforming a workflow model coded in a workflow definition language (WDL) file 10, such as FDL, into a process template 8 implemented in the runtime database 4. The runtime database 4 stores database tables that implement the data structures that provide the status and setup information needed for workflow process execution. Whenever the state of a process activity changes, such information is recorded in the runtime database 4. The runtime database 4 may be implemented using any database program known in the art, such as IBM DB2.**

[0015] The workflow server 6 coordinates and manages the execution of processes for a defined process template 8. The workflow server 6 executes any programs associated with a process defined for the workflow, interprets the process definitions, creates process instances and manages their execution, manages processes and states, logs events,

communicates with users as part of the workflow, etc. The workflow server 6 may include a database client program (not shown) to access and update records related to the workflow being processed maintained in the runtime database 4. The workflow server 6 processing may be distributed across multiple computers to achieve workload balancing.

[0016] The workflow clients 12a, b...n represent the client computers that execute workflow application program interfaces (APIs) to perform workflow related actions and activities and return messages to the workflow server 6. The workflow clients 12a, b...n thus comprise instances of the workflow code on the client computers that allow users to interface with the executing workflow and the workflow server 6. The workflow server 6 would execute activity programs as part of executing the workflow and transmit messages and data to the workflow client 12 to request user action to proceed with the workflow. The actions associated with the nodes and executed by the workflow server 6 may comprise Java servlets. The workflow client 12 may comprise a Web browser capable of executing Java scripts transferred from the Java servlet executing on the workflow server 6. Further, details on implementations and interactions of the workflow server 6 and client 12 are described in the IBM publication "IBM MQSeries Workflow: Concepts and Architecture, Version 3.3", IBM document no. GH12-6285-03 (March, 2001), which publication is incorporated herein by reference in its entirety.

[0017] A workflow builder 20 comprises a system including a buildtime program 22 that implements a plurality of graphical user interface (GUI) panels in which a user may define the components of a workflow model 24. A workflow translator 26 converts the workflow model 24, with the defined workflow components, into a workflow definition language (WDL) file 10 that implements the workflow model 24. The workflow definition language (WDL) may comprise the FlowMark Definition Language (FDL), Workflow Process Definition Language (WPDL) or any other workflow definition language known in the art that is used to define workflows. The workflow translator 24 would transfer the WDL file 10 to the workflow server 6 to transform into a process template 8 in the runtime database 4 in a manner known in the art. Further details of

using the buildtime program 22 to build workflows are described in the copending and commonly assigned patent application "Method, System, and Program for Generating a Workflow", having docket no. STL920000044US1, which application was incorporated herein by reference above.

[0018] The workflow engine 2, and each of the program components therein, such as the runtime database 4 and workflow server 6, may be implemented in one or more computing machines. The workflow clients 12 which provide the workflow interface to users may be implemented on one or more client machines. The workflow builder 20, including the buildtime program 22 and workflow translator 26 programs, may be implemented on one or more computing machines. Any portion of the workflow engine 2, workflow builder 20, and/or workflow client 12, and program components therein, may be implemented on the same computing machines or separate machines. The computing machines used to implement the workflow engine 2, workflow clients 12, and workflow builder 20 may comprise any computing device known in the art, such as a server, workstation, mainframe, personal computer, laptop computer, hand held computer, telephony device, etc.

[0019] One use of a workflow is to generate a final product, which may comprise the result of the effort of a single business unit or the cumulative efforts of multiple users and units within an organization.. To produce the final product, a workflow packet comprised of one or more documents would transfer through various user work stations in the company defined as nodes in the workflow to require the user associated with such node to handle and process and forward to another user to handle. A document is comprised of a multimedia item that has digital content.

[0020] For instance, an insurance company may have to process numerous documents related to an insurance claim, such as photographs, appraisals, expert reports, etc. Employees may spend a substantial amount of time sorting through documents and associating the documents with particular claims. In the workflow model, all the documents related to a single claim would be part of a work packet that may move

through various user stations to review and process. The workflow would comprise the flow of work and actions that are performed on the documents or workflow packet by multiple users in the system.

[0021] The workflow defines the sequence and boundaries of how the work is performed with respect to the documents in the workflow packet, and any restrictions on the order in which documents in the workflow packet must be processed. For instance, before the claim can proceed to a further step, a claims adjuster might be required to ensure that certain documents are included in the workflow packet for the claim before the workflow packet can proceed to further nodes in the workflow, e.g., determining the amount of compensation.

[0022] In workflow terminology, a worklist is a queue of work items. Each work item comprises a unit of work for a node in the workflow that is performed by the users associated with that node. Each work item may be associated with one work packet, which comprises documents or objects that are processed during the work defined for that work item. When a user at one node accesses the work item to perform the work defined therein, that work item is locked, thereby preventing others at that node from accessing the work item.

[0023] A worklist, which is a queue of work for the users of the organization to perform with respect to the workflow packet. The work items within the worklist can be handled by any of the employees/users assigned to the worklist. An action list defines the actions that a user can perform on the work packet objects associated with the work item, such as selections or data that may be entered in the work packet. For example, an adjuster in the claim process workflow can select an option to continue consideration of the claim if it appears valid or select an option to reject the claim. The workflow further consists of the paths defined as the connections between nodes which indicate the order of execution of nodes in the workflow.

[0024] An action list may be associated with a workflow that provides a list the actions that can be invoked at the nodes in the defined workflow. The actions may comprise

programs that are executed at a particular node. In certain implementations, the actions comprise Java methods that the workflow server 6 executes when control proceeds to the node with which the method is associated. Action in the list would be associated with particular nodes. An access list defines a mapping of users that can be assigned to nodes to perform the action associated with such node. An notification feature causes a message to be sent to a specified user if the user associated with a node has not performed the action defined for the node within a specified time frame.

[0025] One or more actions and a user with are associated with the work nodes in the workflow. The work nodes defined for the workflow may comprise a decision point node, collection point node, document node, and assign value node. A decision point node causes the workflow to proceed along a branch of execution based on selection by the user or some other action taken by an external application called at a previous work node. For instance, the path taken to the next node in the workflow may vary if the claim adjuster selects to reject the claim as opposed to approving the claim. A collection point node is a work node where certain documentation is gathered and added to the work packet. The collection node holds and manages work packages that cannot be processed completely until additional information is received. A document node represents a document in the workflow.

[0026] In certain implementations, the workflow model 24 defined using the buildtime program 22 is document centric in that the actions performed at the node concern the processing of work packages that may comprise any content or object that is processed and routed through the workflow. FIG. 2 illustrates the logic performed by the workflow server 6 to execute a workflow. When a user invokes a workflow stored in the runtime database 4, the workflow server 6 accesses (at block 100) the start node of the invoked workflow by interacting with the runtime database 4 in a manner known in the art. From the properties defined for that node, the workflow server 6 determines (at block 102) the actions and user associated with the node. The workflow server 6 further processes (at block 104) the access list defined for the workflow to determine the work item for the

accessed node. If (at block 106) the determined work item currently accessed in the workflow is locked by another user at that node, then the workflow server 6 waits (at block 108) for the lock on the work item(s) to be released. If the work item is not locked or after the lock is released, control proceeds to block 110 where the workflow server 6 places a lock on the determined work item. The workflow server 6 then executes (at block 112) the action associated with the node and communicates data to the workflow client 12 of the determined user requesting user action.

[0027] If (at block 114) notification is enabled for the current node and the deadline has passed (at block 116) without receiving a response from the user, then the workflow server 6 notifies the user specified with the enable notification that the deadline has passed. Upon receiving (at block 118) a response from the user, which may comprise entering information, modifying a work item, adding a work item to the work package, selecting an option, etc., the workflow server 6 unlocks (at block 120) the work item(s) previously locked for the user. If (at block 122) the current node is the stop node, then control ends; otherwise, if there are further nodes to process in the workflow, then the workflow server 6 determines (at block 124) from the path from the current node the next node in the workflow and accesses (at block 126) the next node. Control then proceeds back to block 326 to process the next node.

[0028] The workflow logic of FIG. 2 provides a document centric workflow in that the state of processing work items associated with the node controls the workflow because control cannot proceed to other subsequent nodes that process the locked work item until the node holding the lock completes execution and releases the lock on the work item. Thus, access to work items controls the flow through the workflow.

[0029] With the described implementations, the workflow builder 20 generates a WDL file 10 that may be compatible with workflow engines from different vendors because different vendors may design their workflow engines to be compatible with the WDL format of the WDL file 10. This allows the workflow model defined in the WDL file 10 to be transportable across different vendor workflow engine platforms.

Express Mail No. EL821157938US
Docket No. STL920000098US1
Firm No. 0055.0043

Object Oriented Workflow Architecture

[0030] FIG. 3 illustrates an architecture of object oriented classes and their interrelationship that are used to implement a workflow of nodes. As indicated in the legend 400, a rectangle indicates a class; a line connecting classes indicates an association of the connected classes; a line connecting classes terminating in a filled circle indicates that there may be one or more instances of the class at the end with the circle for each instance of the class at the other end of the line; and a line terminating at a diamond indicates that the class at the diamond end is an aggregate, such that the aggregate object is made up of one or more instances of the class at the other end of the line. FIG. 3 illustrates the relationship of the classes.

[0031] The WorkFlowService class 402 is the starting point for a user wanting to access a workflow. The WorkFlowService class 402 includes methods that allow users to access already defined workflow templates and executing workflows. The WorkFlowService class 402 is associated with the WorkFlowTemplate 404, WorkFlow 406, and WorkFlowList 408 classes. The WorkFlowTemplate class 404 provides methods that allow the user to manipulate workflow process template objects, e.g., process template 8 (FIG. 1), which comprise a defined workflow that is stored in the workflow engine 2. The WorkFlow class 406 provides methods that allow the user to access information and control an executing workflow. The WorkList class 408 includes methods that allow the user to access an executing work list object comprised of work items and information on the current state of the executing work list, i.e., information on work items being processed. The methods in the WorkFlowService class 402 are used to retrieve information on particular workflows, workflow templates, and workflow lists associated with a particular workflow service. The methods from the other classes, such as the WorkFlowTemplate 404, WorkFlow 406, and WorkFlowList 408 classes, can then be used to obtain specific information and control over those workflow templates, workflows, and workflow lists identified by the WorkFlowService class 402 methods.

[0032] The WorkFlowTemplate class 404 provides information on a workflow template. A workflow object from the WorkFlow class 406 represents an executing workflow. The WorkFlowContainer class 410 includes methods to instantiate a container object that includes information on one container used to transfer data between nodes. Users at nodes may access data in the container and update the container with additional data. The data in the container may be used by the action being executed at a node. The WorkFlow class 406 is associated with the WorkFlowNotification class 412, which is used to provide notifications, such as notifications if a user does not perform an action at a node within a predefined time period. There may be many notifications provided for one workflow. The WorkFlow class 406 is further associated with the WorkFlowItem class 414, such that one executing workflow may be associated with one or more work items indicating a unit of work to perform for a node within the workflow. The WorkFlowItem class 414 is associated with the WorkFlowContainer class 410, such that one container may be used at a work item to provide data to the user executing the unit of work defined by the work item. The relationship between the WorkFlow class 406 and the WorkFlowItem class 414 indicates that there may be many work item objects associated with one executing workflow. The class architecture of FIG. 3 further illustrates that a workflow list of the WorkFlowList class 408 is an aggregate of the workflow from the WorkFlow 414 Item class and workflow notifications from the WorkFlowNotification 412 class.

[0033] The above object oriented architecture of FIG. 3 defines how the different classes interrelate in order to implement a workflow. Each of the above interrelated classes 402, 404, 406, 408, 410, 412, and 414 provides interfaces/methods that may be used within a workflow computer program to implement the workflow and actions performed at a node. The workflow program would be executed by the workflow server 6 (FIG. 1) in the workflow engine 2.

[0034] Following are examples of some methods of the WorkFlowService class 402, including:

WorkFlowService(): constructs a new workflow service, which provides access to different workflow services in the workflow engine 2 (FIG. 1). Each workflow service is associated with workflow templates, executing workflows, and workflow lists of work items for a workflow.

connect: provides a user name, authentication, and connection string to use to authenticate a user to provide access to a requested workflow service, which allows access to workflow templates, work lists, etc.

connection: handle returned to a user to allow access to a particular workflow service.

setDatastore: a reference to a data store including documents and objects used by the work items in the workflows associated with the workflow service. Thus, different workflows for a workflow service may process documents within workflow packages from the same data store.

listWorkFlows: returns a list of all workflow objects of the WorkFlow class 406.

listWorkLists: returns a list of all work list objects of the WorkFlowList class 408.

listWorkFlowTemplates: returns a list of all template objects of the WorkFlowTemplate class 404.

[0035] Following are examples of some methods of the WorkFlowService class 402, including:

WorkFlowTemplate(): constructs a workflow template object including a defined workflow. This workflow template may be created using the GUI panels and buildtime program described above.

name: returns name of a workflow template.

description: returns a description of the work performed by a workflow template.

modifiedTime: time the workflow template was last modified.

[0036] Following are examples of some methods of the WorkFlow class 406, including:

WorkFlow(): constructs a workflow object representing a workflow comprised of nodes and work items for a specified workflow. The workflow may also be provided a container that is used to allow users of different work items to communicate and/or a work packet comprised of one or more documents or objects to be processed as part of the workflow.

get/setName: returns or sets the name for a workflow.

workFlowTemplateName: returns the name of the workflow template associated with the workflow.

notificationTime: returns the time of the last notification generated for the workflow in response to a user not performing an action for one accessed node within a specified time period.

modifiedTime: Returns the last time the workflow was modified.

stateChangeTime: returns the last time a state change occurred with the workflow:

startTime: returns the time the workflow was started.

endTime: returns the time the workflow ended.

state: returns a state of the workflow, such as ready, running, finished, terminated, suspended, terminating, suspending, deleted, etc.

inContainer: returns the input container associated with the workflow.

start: starts a workflow with a container if the state is ready.

terminate: terminates the workflow if the state is running, suspended, or suspending.

suspend: suspends the workflow if the state is running.

resume: resumes a suspended workflow if the state is suspended and suspending.

add: adds a workflow to the system that is associated with one specified workflow template.

Express Mail No. EL821157938US
Docket No. STL920000098US1
Firm No. 0055.0043

[0037] Following are examples of methods of the WorkFlowContainer class 410, which instantiates a container object used with a workflow to transport information among the nodes.

WorkFlowContainer(): constructs a container object for a container used within a particular workflow.

get/setPriority: get/sets the priority for an item in the container.

get/setActivityNode: get/sets the current node being processed, may also get/set information on the current activity node.

get/setWorkPacketID: get/sets an identifier of a work packet being routed through the system.

get/setActionPerformed: get/sets information on an action being performed.

get/setUserVariable: get/sets a variable maintained in the container, that may have predefined values. The priority is maintained for a user variable in the container.

retrieve: retrieves and refreshes the container.

update: updates the container data.

[0038] Following are examples of some methods of the WorkList class 408, where a work list object is a representation of a work list in the system. As discussed, a work list object comprises a collection of work items and notifications for an executing workflow.

WorkList(): constructs a work list object for a specified work list. A work list consists of work items.

get/set ACLName: get/sets the action control list (ACL) name for the work list including the actions that may be performed as part of units of work for the work list.

listWorkItems: lists the work items on the work list.

listWorkItemsByTemplate: returns the work items for the work list by the specified workflow template name.

listWorkItemsByNode: returns a list of the work items assigned to each node in the work flow.

listProcessNotifications: lists notifications generated during workflow that are associated with the workflow process. For instance, the notification provides a general notification for the workflow. In certain implementations, a notification process is activated and performed as a background process to generate notifications.

listActivityNotifications: lists notifications generated during workflow that are associated with a particular activity, such as a user not performing an activity within a specified time. For instance, the notification may enable notifications for activities at particular nodes.

add/update/delete/retrieve: separate commands that allow user to add, update, delete, and retrieve a work list.

[0039] Additional commands may be provided to access the information in the work list, such as filter commands to provide filters for accessing information from the work list, thresholds of the number of items that can be in the work list, etc.

[0040] Following are examples of some methods of the WorkFlowItem class 414, where a work item object represents a unit of work performed in the workflow. The following methods are used to create and modify work items, and obtain information thereon.

WorkFlowItem(): constructs a work item for a specified workflow, node, and owner.

name: returns the name of the node to which the work item is assigned.

state: returns a state of the work item, such as not set, ready, running, finished, terminated, suspended, disabled, checked out, in error, executed, etc. A work item is checked out when a user has accessed the work item to perform the actions defined for the work item.

workFlowName: returns the name of the workflow including the work item.

workFlowTemplateName: returns the name of the workflow template including the work item.

priority, owner, notificationTime, startTime, creationTime, modifiedTime: methods that return information on the priority, owner, time of last notification, time of creation and time of last modification for a work item, respectively.

retrieve, start, finish: methods used to retrieve, begin executing, and complete a work item, respectively.

checkIn, checkOut: checkOut locks a work item to prevent other users at a node from accessing the work item and changes the state of the work item to checked out. Upon check out, the container associated with the work item is accessed from the previous node using the inContainer method. The checkIn method receives the completed work item from the user, releases the lock, and provides the container to route to the next node.

inContainer: method that obtains container from previous node for use with work item checked out at current node being processed.

outContainer: method generates an out container to include contents of container user accessed at work item, including any changes made by the user to the data in the container. A handle of the out container is generated and provided with checkOut method called for the next node to provide that container to the user of the next node in the workflow.

[0041] Following are examples of some methods of the WorkFlowNotification class 412, where a notification object represents a generated notification. The following methods are used to create and modify notifications, and obtain information thereon.

WorkFlowNotification(): constructs a notification object having a specified notification name, notification type, and owner name for a specified workflow service and workflow. The notification type indicates how the owner is notified.

state: returns a state of the notification, such as not set, ready, running, finished, terminated, suspended, disabled, etc.

priority, owner, notificationTime, startTime, creationTime, modifiedTIme, receivedTime: these methods return the priority of the notification, owner of the notification, time that must elapse before the notification is generated, time the notification started, time the notification was crated, time of last notification to the notification, time the notification was received, respectively. The notification would be started and executed as a background process.

receiveReason: returns a received reason for the notification.

retrieve, cancel: methods that retrieve and cancel a notification, respectively.

transfer: transfers a notification to a specified user. In this way, a notification can be transferred from the current owner to some other user.

[0042] The above described methods and classes would be included in a workflow program executed by the workflow server 6 (FIG. 1) to execute the workflow. The methods described above would be used to access and modify the workflow related objects, such as the workflow, work items, notifications, containers, etc. when running the workflow. The above described methods may also be used in other programs that can obtain information and status on a workflow.

[0043] FIGs. 4-5 illustrate an example of program logic in a workflow program executed by the workflow server 6 (FIG. 1) utilizing the above discussed methods to implement a workflow. With respect to FIG. 4, control begins at block 450 where the program calls the constructor methods, WorkFlowService() to construct a workflow service object. The workflow program would then call (at block 452) the WorkFlowService list methods, such as listWorkFlows, listWorkLists, listWorkFlowTemplates, to obtain information on the workflows, workflow templates, and work lists for a workflow service. This information may then be presented to a user for selection. Various other methods in the classes may be called to access information

on the workflow to present to the user when making a decision on which workflow to execute.

[0044] At block 454, user selection of a workflow to process is received. The workflow program then calls (at block 456) the WorkFlow start method to start the workflow. The workflow program then calls (at block 458) the listWorkItemsByNode method to obtain all the work items for the started workflow, and the nodes to which the one or more items are associated. The workflow program then performs a loop at blocks 460 through 490 for each node *i* in the workflow, as determined from the list of work items by node. For each node *i*, the workflow program performs a loop at block 462 to 488 for each work item *j* associated with node *i*. If (at block 464) there is a notification for the work item and the user that is the owner of the item, as determined from the methods, then the workflow program retrieves (at block 466) retrieves the notification and then starts a monitor to determine if the time period for the notification has elapsed without the work item completing. From block 464 or 466, the workflow program calls (at block 468) the checkOut method to lock the work item *j*. The inContainer method is called (at block 470) to access any container associated with the work item *j*. Once the work item *j* is locked, the workflow program then executes (at block 474) the actions associated with the work item *j*.

[0045] Control then proceeds to block 476 in FIG. 5, where the workflow program calls container get and set methods to access or modify the data and variables in the container accessed for the work item *j* in response to executing actions assigned to that work item *j*. For instance, as part of performing actions for a work item, the user of the work item may read and write data to the container. The workflow program receives (at block 482) indication from a user that the actions associated with the work item have completed. The workflow program further calls (at block 486) the checkIn method to release the lock on the work item *j* and the outContainer method to generate a new container including any updates to provide to the user at the next node in the workflow. The handle to the new container would be used in the next called checkOut method to provide the container

to the user at the next node of the workflow. If there are further work items for the node *i*, then control proceeds (at block 488) back to block 452 to retrieve the next work item. After completing all the work items for node *i*, control proceeds (at block 490) back to block 460 to process the next node in the work list.

[0046] The above described logic utilized workflow related classes and the methods therein to implement a workflow and obtain information thereon. The workflow server 6, or some other component in the workflow engine 2 (FIG. 1), would then translate the workflow objects and methods into application specific commands, such as Structured Query Language (SQL) commands to manipulate the data in the runtime database 4 and process template 8 to obtain information on the workflow and implement workflow operations.

Using Database Stored Procedures In the Workflow Design

[0047] FIG. 6 illustrates a workflow architecture including a client 500, such as the workflow clients 12a, b...n described with respect to FIG. 1. The client 500 is capable of invoking, in response to user input or executing an application program, one or more workflow application programming interfaces (APIs) 502 to perform workflow related operations and workflow metadata APIs 504 to access, update or modify workflow related information maintained in a database 506. The workflow APIs 502 may implement the object oriented workflow classes described above, such as those described with respect to FIG. 3. In certain implementations, the workflow APIs 502 and 504 may be implemented in a commonly used object oriented programming language, such as Java.

[0048] The client 500 further includes a mapping 508 of the APIs 502 and 504 to stored procedure calls that invoke stored procedures 510a, b...n , 512a, b...n on a database server 514. Thus, for each API 502 and 504, there are one or more stored procedure calls to invoke one or more stored procedures 510a, b...n, 512a, b...n on the database server 514 to perform the action specified by the API command. The client stored procedure call

corresponding to the called API provides input parameters to the stored procedure 510a, b...n, 512a, b...n. In response to the call, the stored procedures 510a, b...n, 512a, b...n execute within the database server 514 and may processes numerous database records in the database 506 according to the input parameters or perform a non-SQL related action, such as transmitting native workflow code to the workflow server 518 to execute. In certain stored procedure implementations, the client cannot interrupt the stored procedures during execution. The stored procedures 510a, b...n, 512a, b...n may comprise a block of procedural constructs and may include Structured Query Language (SQL) statements, i.e., an application program. Stored procedures are maintained at the database server 514 for access and reuse by multiple clients 500. Further details of stored procedures are described in the publication "A Complete Guide to DB2 Universal Database," "A Complete Guide to DB2 Universal Database," which was incorporated by reference above.

[0049] The database server 512 includes a database program 516 to perform database operations, such as execute stored procedure calls and execute SQL statements to access data in the database 506. The workflow function stored procedures 510a, b...n include the workflow engine native code, e.g., C, C++, etc., capable of causing a workflow server 518, such as the workflow server 6 described with respect to FIG. 1, to perform the workflow function specified by the corresponding workflow API 502. The workflow metadata stored procedures 512a, b...n include SQL statements to access data in the database 506 and perform database operations on such data.

[0050] The client 500 executing the workflow APIs 502 and 504 may comprise a separate client machine that communicates with the database server 514 over a network or other connection (not shown), or comprise a program executing on the database server 514. The database server 514 comprises a separate server class machine for maintaining the stored procedures 510a, b...n and 512a, b...n and database program 516 that controls access to the database 506. The database program 516 may comprise any database

client/server program known in the art, such as IBM DB2, Oracle Corporation's ORACLE 8, Microsoft SQL Server, ** etc.

[0051] In Java implementations, the mapping 508 may comprise Java Database Connectivity (JDBC) CallableStatements, where each CallableStatement assigns a stored procedure call in the native SQL of the database program 518 to an escape syntax, which may comprise the workflow APIs 502, 504. Thus, for each API, one separate JDBC CallableStatement would define a stored procedure to call to implement the workflow related operation associated with the API.

[0052] FIG. 7 illustrates logic implemented in the workflow APIs 502 and 504 in the client 500 to implement the client 500 invoked API. Upon receiving (at block 550) an API call 502, 504, the mapping 508 is processed (at block 552) to determine one or more stored procedure calls corresponding to the received API. The client 500 then invokes (at block 554) the determined stored procedure call(s), including the name of the called stored procedure and the database server 514 including the called stored procedure, and transfers the call to the database server 514. In the described implementations, the client 500 generates a stored procedure CALL statement specifying the name of the server stored procedure 510a, b...n, 512a, b...n and the parameters received as input by the stored procedure 510a, b...n, 512a, b...n.

[0053] The mapping 508 may associate multiple stored procedure calls with one API call, such that invocation of the API call invokes the associated multiple stored procedures.

[0054] FIG. 8 illustrates logic implemented in the database program 516 to process the stored procedure calls from the client 500. In response to receiving (at block 560) the stored procedure call, communicates from the client 500 to the database server 514, the database program 516 executes (at block 562) the called stored procedure 510a, b...n, 512a, b...n. If (at block 564) the received API is a workflow API 502, then the corresponding called stored procedure 510a, b...n includes the workflow server 518 native code. This stored procedure 510a, b...n would then instruct the database program

516 to transfer (at block 566) the native code included in the called stored procedure to the workflow server 518 to execute and perform the function defined by the API 502. With the workflow function stored procedures 510a, b...n the workflow server 518 native code is wrapped in the stored procedure. Otherwise, if the stored procedure is a workflow metadata stored procedure 512a, b...n including SQL statements, then the database program 516 would execute (at block 570) such SQL statements to access workflow metadata in the database 506 and perform any specified SQL operations thereon. After the executed stored procedure completes all processing, status or result data may be returned to the client 502.

[0055] There are numerous ways the stored procedures may be used to access and update workflow metadata in the database 506 and then invoke workflow operations on the workflow server 6. For instance, during workflow runtime, the client 500 may call workflow metadata APIs 504 to access workflow related information, such as worklists, access control lists, etc., and then subsequently call a workflow API 502 that specifies a workflow function to perform in the workflow server 518 on the metadata accessed from the database 506. Alternatively, a stored procedure 510a, b...n or 512a, b...n may include SQL statements to access data from the database 506 and then transfer such accessed metadata along with native code to the workflow server 518 to execute. For instance, the workflow function stored procedure 510a, b...n may access worklists, access control lists, etc., from the database 506 and then pass such data along with native code to the workflow server 6 to process during workflow operations. Still further, one stored procedure 510a, b,...n, 512a, b...n may access workflow metadata from the database 506 and then call and directly pass such accessed workflow metadata to a workflow function stored procedure 510a, b...n to pass to the workflow server 518 to process.

[0056] Still further, a stored procedure may pass native code to the workflow server 518 to execute and receive return workflow metadata from the workflow server 518 to apply to the database 506. The same or another stored procedure on the database server 514 may then use SQL statements to update the database 506 with the returned workflow

metadata. Alternatively, the returned workflow metadata may be passed back to the client 500, and the client 500 may further call workflow metadata APIs 504 to invoke a workflow metadata stored procedure 512a, b...n to update the database 506 with the returned workflow metadata from the workflow server 518.

[0057] Thus, any workflow related operation comprising a combination of calls to the workflow server 518 and accesses of workflow metadata in the database 506 may be combined into one stored procedure in the database server 514 or distributed across multiple stored procedures.

[0058] Moreover, in certain implementations, one workflow API 502 may map to multiple stored procedure calls. For instance, one stored procedure call authenticates the user and provides a connection to the workflow server 518 and runtime database 516, one other stored procedure performs the requested workflow related action, and yet a further stored procedure disconnects the authenticated user.

[0059] The described implementations transform client API calls to stored procedure calls that are executed on a separate database server to perform the function specified by the API call with respect to the workflow server 8 and/or the database 506.

[0060] In implementations where multiple vendors provide workflow engines for use in a federated system, each vendor may use the same API calls, but provide a different set of stored procedure code to implement the action specified with the API to accommodate the differences in the vendor's workflow server and database structure. Thus, the workflow APIs 502, 504 remain the same across vendor implementations, whereas each vendor provides the specific stored procedure programs 510a, b...n and 512a, b...n to perform the action associated with the APIs on the vendor specific workflow server 518 and database 506.

Additional Implementation Details

[0061] The preferred embodiments may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software or code. The term "article of manufacture" as used herein refers to code or logic implemented in a computer readable medium (e.g., magnetic storage medium (e.g., hard disk drives, floppy disks, tape, etc.), optical storage (CD-ROMs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which preferred embodiments are implemented may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without departing from the scope of the present invention, and that the article of manufacture may comprise any information bearing medium known in the art.

[0062] The workflow client and server may be implemented within any vendor workflow program known in the art.

[0063] In the described implementations, the actions were implemented as Java methods. Alternatively, the actions may be implemented in any programming language known in the art.

[0064] In the described implementations, the class architecture is implemented as an object oriented class architecture. Alternatively, non-object oriented programming techniques may be used to implement the described class architecture.

[0065] The foregoing description of the preferred embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications

and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

**MQSeries, IBM, and DB2 are registered trademarks of International Business Machines Corp.; Microsoft and Microsoft SQL Server are trademarks of Microsoft Corporation; ORACLE is a trademark of the Oracle Corporation; Java is a trademark of Sun Microsystems, Inc.